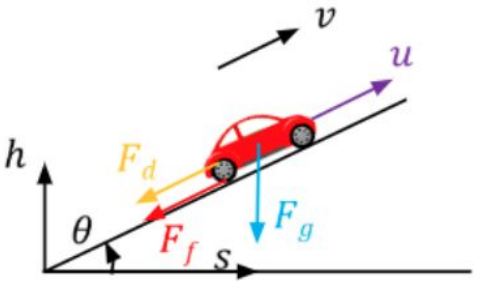
Cruise Control

By: Siddhik Reddy Kurapati

Content

1) Problem and approaches
2) Approach - Trajectory Planner + Control Approach
3) Linear Feedback and feedforward approach(Control Approach)
4) Model Predictive Control approach(Control Approach)
5) PID approach (Control Approach)
6) Trajectory Planner
7) Model Implementation
8) Results
9) Appendix to run the Simulation files

Cruise Control Problem



Problem has been divided into 2 subproblems:

1)Trajectory Planner

2)Control for reference/setpoint tracking

$$m\dot{v} = u - mg\sin(\theta) - F_d - F_f$$

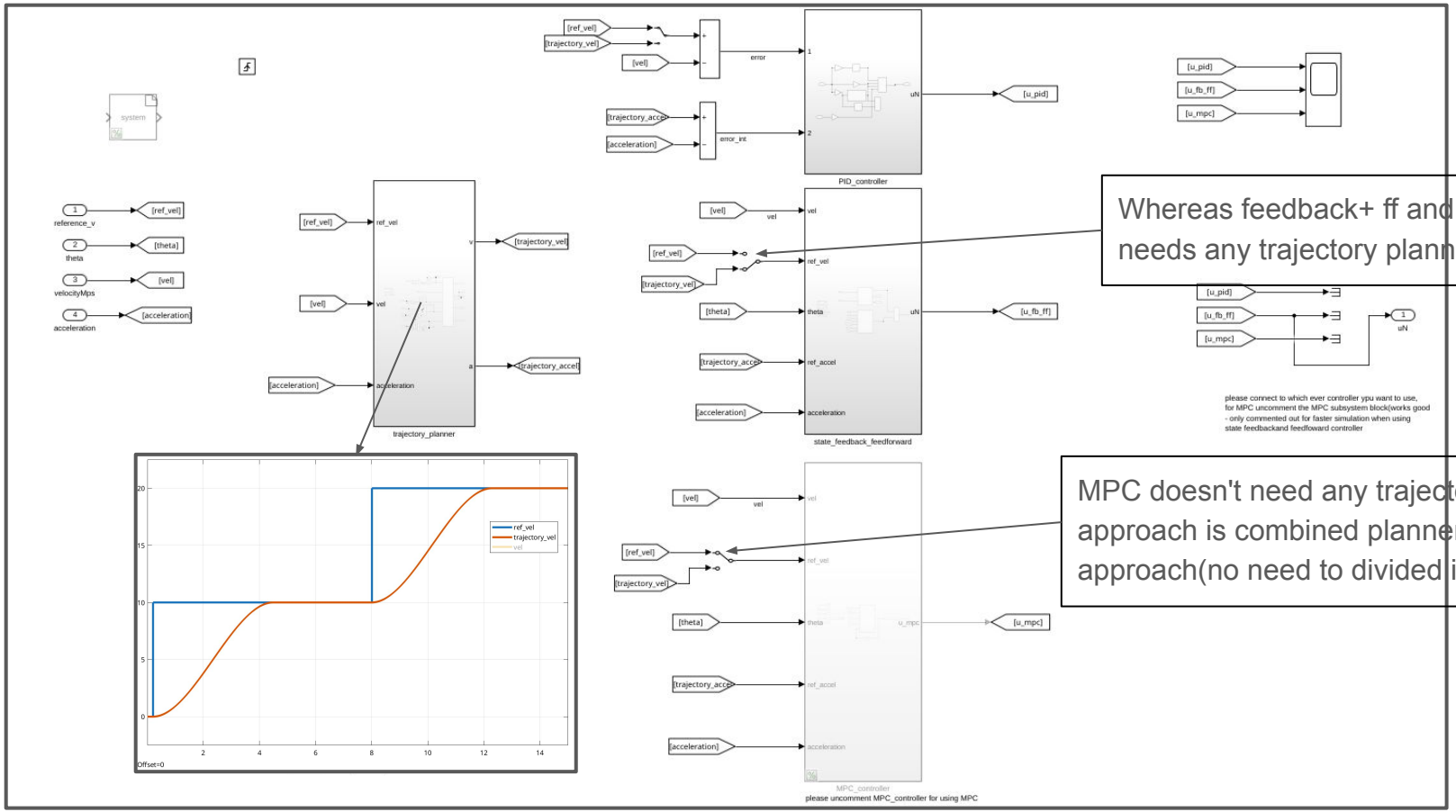
$$F_d = \frac{1}{2}C_d\rho A v^2 \cdot sgn(v)$$

$$F_f = mg\mu\cos(\theta) \cdot sgn(v)$$

Inputs: theta, velocity setpoint

Assumptions: m(2500 kg) is the vehicle mass, Cd(0.28) is the coefficient of drag, ρ (1.225 kgpm3) is the air density, A(2.5 m2) is the drag area, μ(0.7-dry) is the coefficient of friction, gravity(9.8 mps2) are assumed to be constant.

Trajectory Planner + Control Approach



Whereas feedback+ ff and pid approach needs any trajectory planner.

MPC doesn't need any trajectory planner, this approach is combined planner and control approach(no need to divided in 2 sub-problem)

Feedback and ff approach

Model :: $\dot{v} = f\big(v(t), u(t), p(t)\big) = \frac{u}{m} - g\sin(\theta) - \frac{1}{2m} C_d \rho A v^2 . \mathrm{sgn}(v) - g\mu\cos(\theta) . \mathrm{sgn}(v)$

Feedback Law:: $u = k(x - x_c) + r$, r-> can used to include feedback term also.

Linearizing about operating point v_c, u_c)(can be equilibrium point)

Assumptions: For pole placement: λ=-0.9*5

$$\dot{v} = f\big(v(t), u(t)\big)$$

$$\dot{v} = f(v_c, u_c, t) + \frac{\partial f}{\partial v}\Big|_{\{v_c, u_c\}} (v - v_c) + \frac{\partial f}{\partial u}\Big|_{\{v_c, u_c\}} (u - u_c)$$

$$\lambda v = f(v_c, u_c, t) + A\Big|_{\{v_c, u_c\}} (v - v_c) + B\Big|_{\{v_c, u_c\}} (k(v - v_c) + r - u_c)$$

Into 2 equations(pole placement):

$$\lambda v = A\Big|_{\{v_c, u_c\}} (v) + B\Big|_{\{v_c, u_c\}} (kv) \rightarrow k = \frac{\lambda - A}{B}$$

$$\mathrm{r} = -f(v_c, u_c, t) + A\Big|_{\{v_c, u_c\}} (v_c) + B\Big|_{\{v_c, u_c\}} (v_c + u_c)$$

$u(k) \mathrel{+}= k_i \sum e_2(k) + k_p e_2(k)$,
Where $e_2(k) = a(k) - a_{ref}(k)$

PID approach: $u(k) = k_i \sum e_1(k) + k_p e_1(k) + k_d\big(e_1(k) - e_1(k-1)\big)$, where $\mathrm{e_1(k)} = \mathrm{v(k)} - \mathrm{v}_{ref}(k)$

Model Predictive Control approach

Cost Function:

$$J = \sum_{k=0}^{N-1} e_k^T Q e_k + u_k^T R u_k + e_N^T P e_N,$$

where $e_k = [v_k - v_s], R \geq 0, for\ k = 0\ to\ N-1$

Subject to:

$$\dot{v} = f\big(v(t), u(t), p(t)\big)$$

$$\dot{v} = \frac{u}{m} - g\sin(\theta) - \frac{1}{2m} C_d \rho A v^2 . \mathrm{sgn}(v) - g\mu\cos(\theta) . \mathrm{sgn}(v)$$

$$a_{min} * dt \leq v(k+1) - v(k) \leq a_{max} * dt$$

Model mismatch issues: Introduce integrator state $\frac{\Delta int_k}{dt} = K_i * \frac{i - i_{ref}}{dt}$

Delay compensation issues: 1) Modeling additional state u_{k-d} or 2)smith predictor

For (behavior) trajectory modeling: acceleration as state $jerk_{min} * dt \leq a(k+1) - a(k) \leq jerk_{max} * dt$

Assumptions: a_min = -3.5m/s^2 and a_max = 3.5m/s^2

Trajectory Planner

Trajectory planning equation:: $v(t) = a_0 + bt + ct^2 + dt^3$

Acceleration equation:: $a(t) = b + 2ct + 3dt^2$

Constraints:

$$1)\ v(0) = a_0$$

$$2)\ v(t_f) = a_0 + bt_f + ct_f^2 + dt_f^3$$

$$3)\ a(0) = b$$

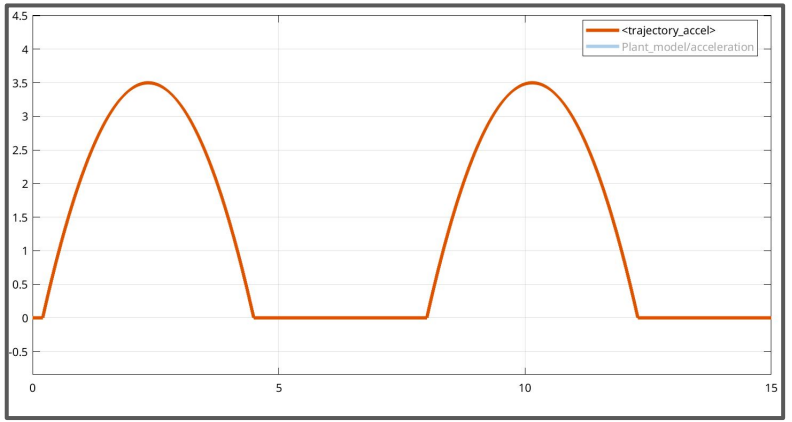
$$4)\ a(t_f) = b + 2ct_f + 3dt_f^2$$

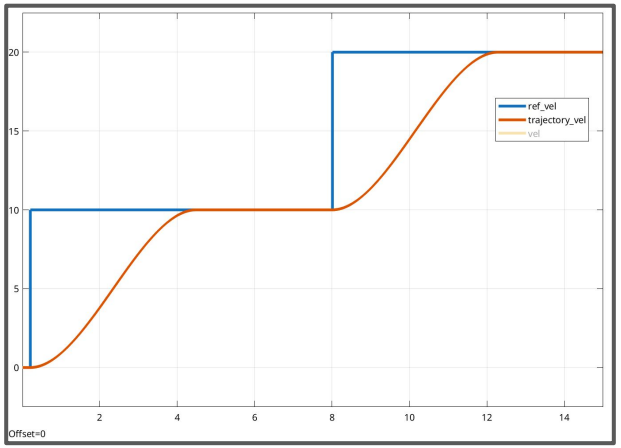
5) $v(tf) > v(0)$, $\quad a_{max} = a(t_{mid}) = b + 2ct_{mid} + 3dt_{mid}^2, t_{mid} = \frac{t_f}{2}$

5) $v(tf) < v(0)$, $\quad a_{min} = a(t_{mid}) = b + 2ct_{mid} + 3dt_{mid}^2, t_{mid} = \frac{tf}{2}$

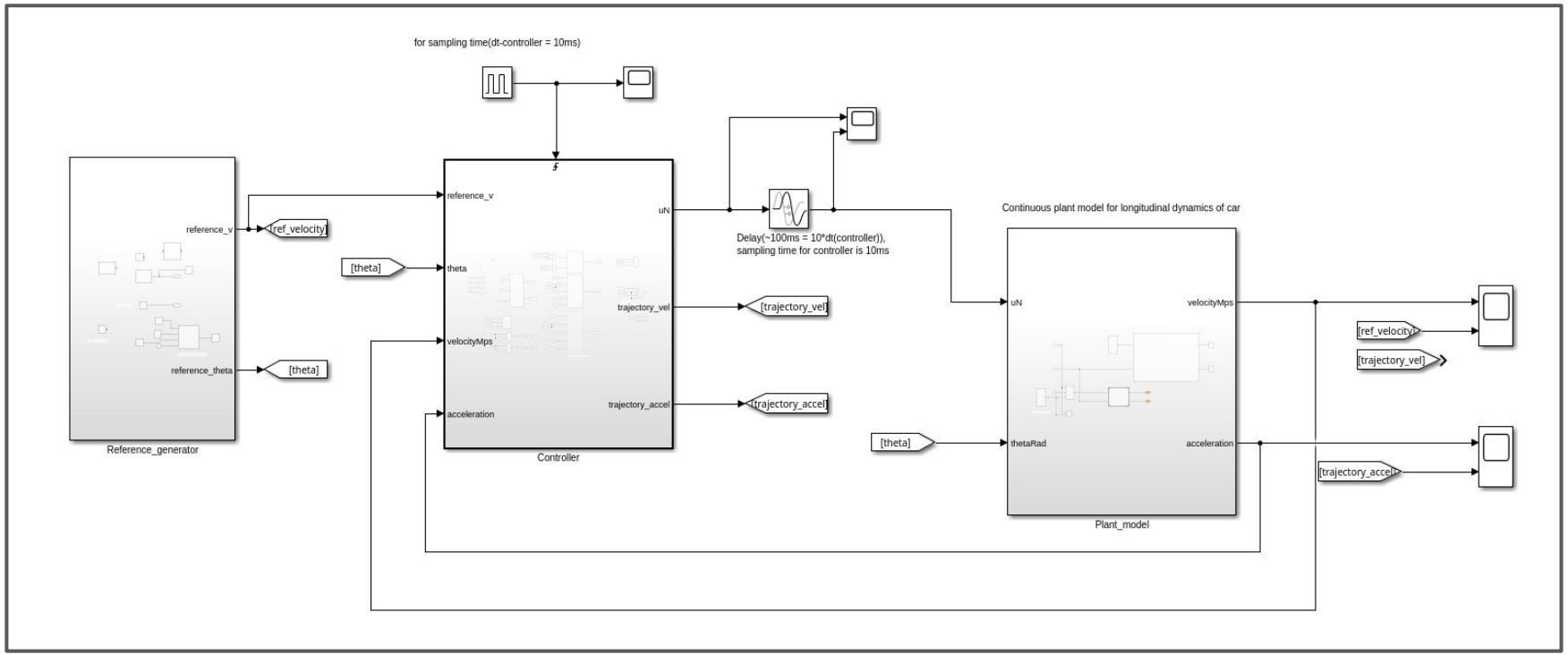
Assumption : Velocity has cubic trajectory profile

We can use a trapezoid trajectory/ b-spline/beizer/nth order polynomial



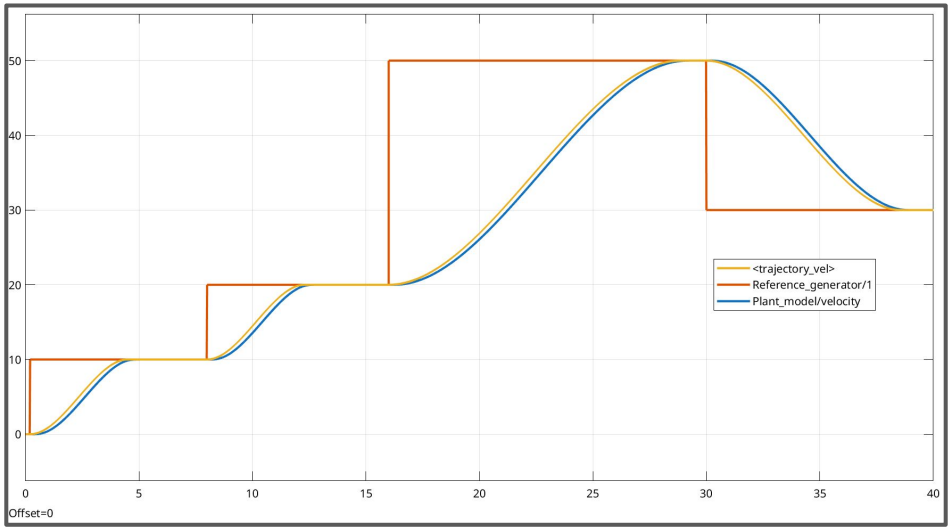


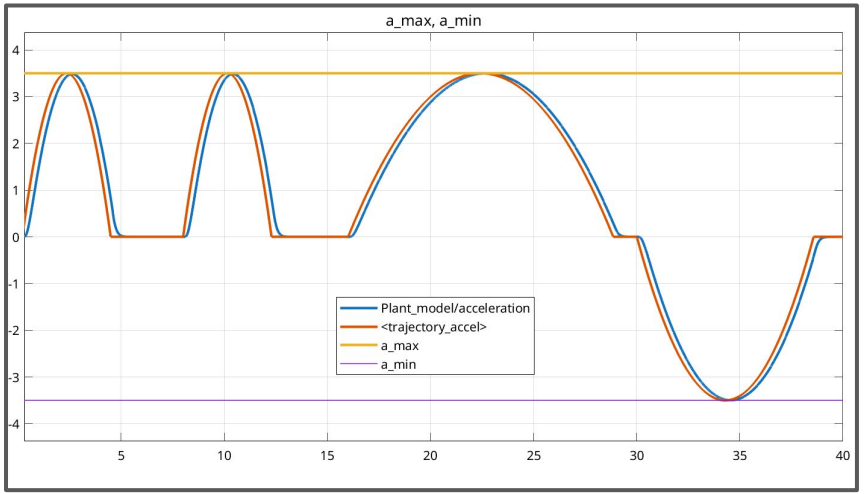
Model Implementation



Assumptions :
Sampling time(dt) = 10ms(Controller)
Delay - 100ms(~10*dt)
Theta as input to the model.

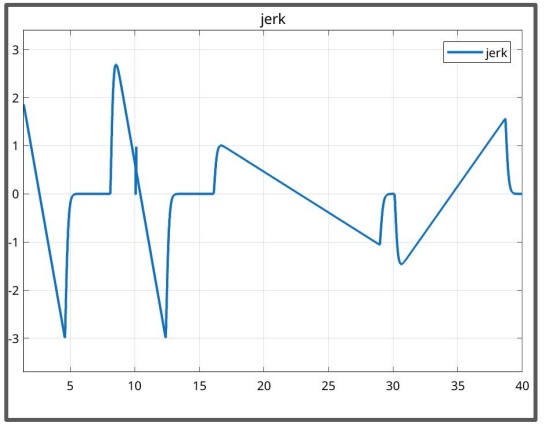
Results - FB+FF



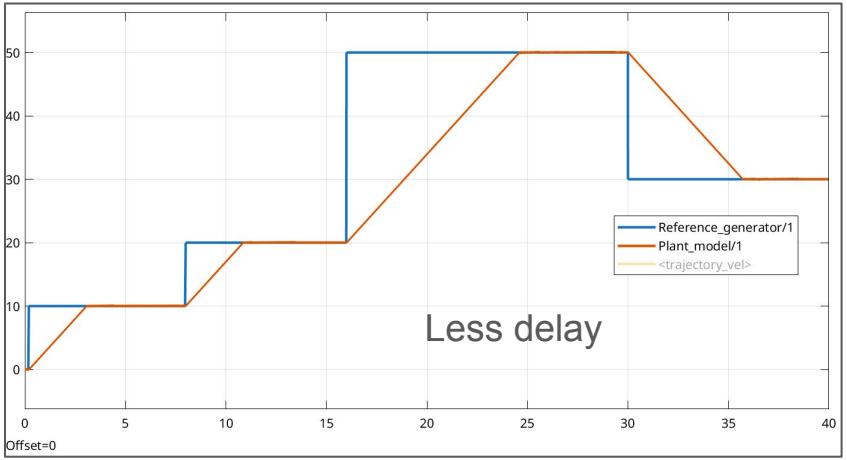


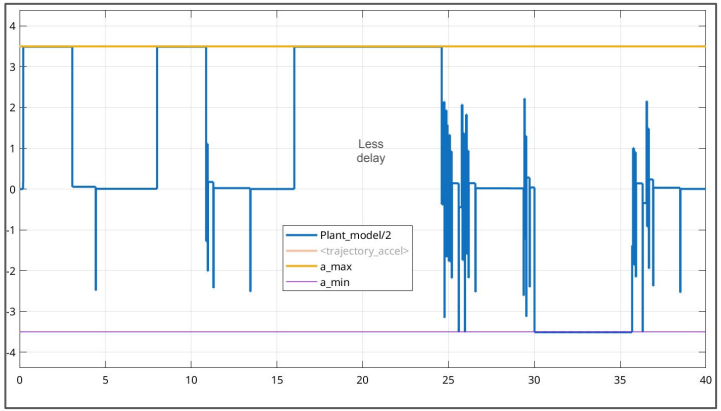
Control approach:(Both (FB+FF) and MPC has good tracking performance

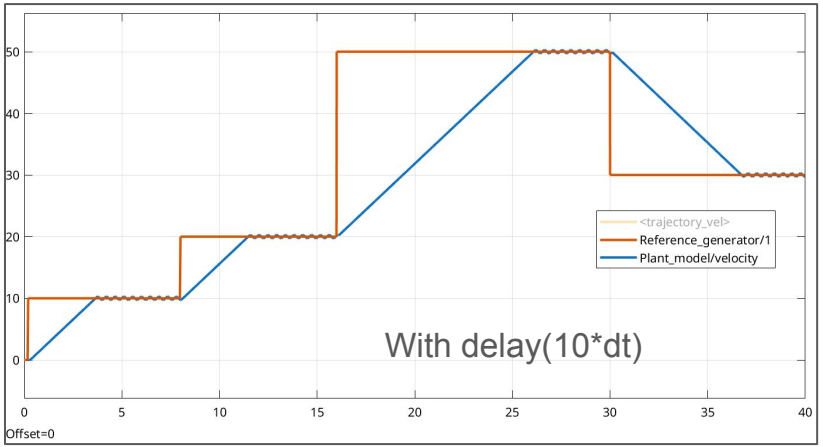
Acceleration are within ub(3.5m/s2) and lb(-3.5m/s2) of accelerations limits



MPC results







- Even with 10*dt delay, mpc tracking is good but with slight oscillation(which can reduced by number of like delay compensation/ increasing the horizon/ tuning cost matrix etc~refer to mpc slide)
- As mentioned in previous slide, MPC doesn't need any trajectory planner, this approach is combined planner and control approach(no need to divided in 2 sub-problem)
- We can add additional states like integrator states to avoid model mismatch issues or accelerations states for profile modification etc

Improvements

1. For fb and ff, Introduce a advanced state machine for trajectory planner (to get behavior etc).
2. Since the fb and ff approach relies on model parameters, if model parameters aren't accurate then maybe introduce more tuning values for control/tracking performance for uncertain models.
3. For mpc approach , We can add additional states like integrator states to avoid model mismatch issues or accelerations states for profile modification etc.
4. Switching to lookup based or neural net approach for parameters.
5. Adding filters if the sensors aren't as accurate.
6. Finally, we can change this to Adaptive cruise control, with using FSM, observers, filters etc.

Thank You

Appendix

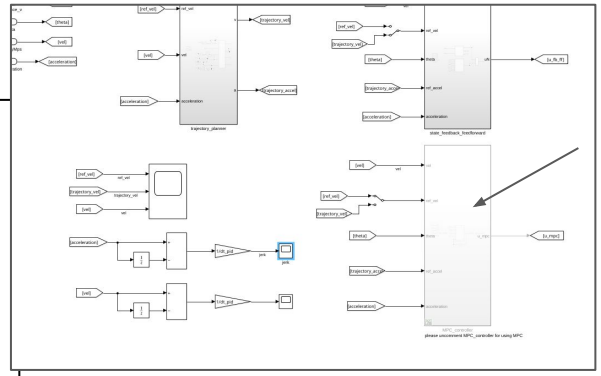
- Matlab /Simulink(2024b) is required.
- For mpc (additional libraries/tools like optimization toolbox is required. So mpc is commented out so it won't cause any compilation issues if the toolbox is not present. So, you can check out responses of other control inputs.

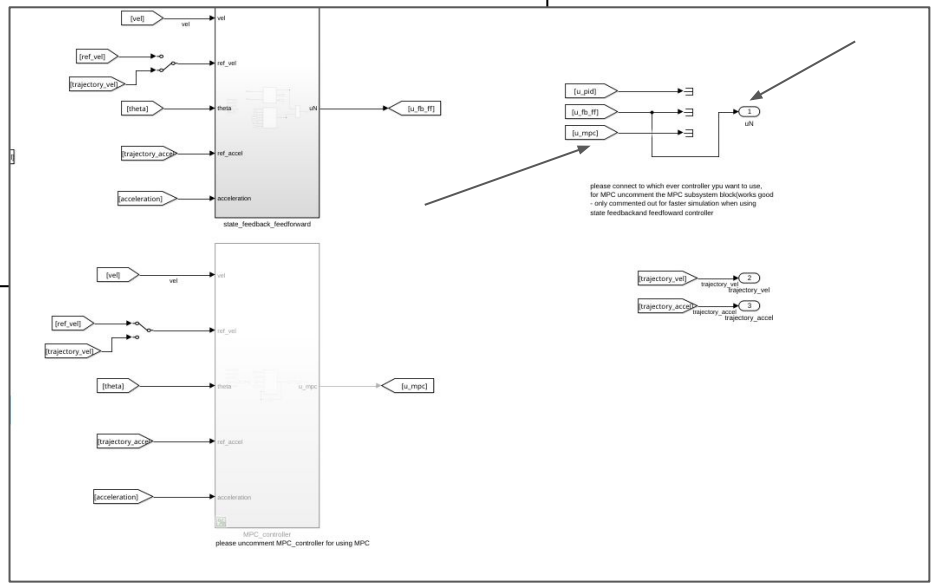
- Solutions zip has to be extracted into single folder.
- For running main.slx is required, initial.m script is automatically initialized and simply need to run the simulink(main.slx) file.
- After running, you can check scopes in outermost layer and also inside the controller block(additional scopes are present in the subsystem)

- For MPC simulations(If the optimization toolbox is present), then proceed to next slide

Appendix

- For running MPC simulation, we simply need to uncomment to subsystem block in controller block
- And, disconnect u_fb_ff and attach it to u_mpc for controller output from mpc for uN signal output.





Different MPC formulation

Generic formulation

$$J = \sum_{k=0}^{N-1} l(x_k, u_k) + l_f(x_N)$$

Subject to: $x_{k+1} = f(x_k, u_k)$

$$g(x_k, u_k) \leq 0$$

$$h(x_k, u_k) = 0$$

$$x_{min} \leq x_k \leq x_{max}, \quad k = 1,....,N$$

$$u_{min} \leq u_k \leq u_{max}, \; k = 0,..,N-1$$

x_0 = current state

- Solved by SQP/QP solvers, more accurate optimal solutions, computationally expensive.

Linear MPC formulation

$$J = \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P x_N$$

Subject to: $x_{k+1} = Ax_k + Bu_k$

$$Cx_k + Du_k \leq 0$$

$$Ex_k + Fu_k = 0$$

$$x_{min} \leq x_k \leq x_{max}, \quad k = 1,....,N$$

$$u_{min} \leq u_k \leq u_{max}, \; k = 0,..,N-1$$

x_0 = current state

- Solved by QP solvers, fast evaluations, suitable for systems where linear approx. are valid.

Benefits

- Handles Multivariable(MIMO) Systems: It naturally optimizes control action across all variables simultaneously.
- Constraint Handling: Incorporates constraints(actuator limits, safety ranges), ensures feasible and safe operation.
- Predictive capability and Optimization-based: Forecasting future states and control decisions can be optimized at each time step.

Challenges

- Computationally expensive: increasing complexity, MPC's can be computationally demanding.
- Model accuracy: MPC heavily relies on plant/system model dynamics for prediction step.
- Robustness: Linear MPC or NMPC isn't inherently robust to model disturbances and uncertainties.

Integrator Selection

- Integrators are used to approximate integration in discrete-time(numerical) systems from Continuous Plant Model : $\dot{x} = f(x(t), u(t), p(t))$

Explicit Integrator

- Computes the next value using only known/current information.

$$x_{n+1} = x_n + h.f(t_n, x_n)$$

- Pros:
- Simple, Fast and easy to implement.

- Cons:
- Becomes unstable if sampling time is large.
- Forward and backward Euler methods are used to solve.

Implicit Integrator

- Computes the next value using only known/current information.

$$x_{n+1} = x_n + h.f(t_{n+1}, x_{n+1})$$

- Pros:
- Much more stable even with larger time steps.

- Cons:
- More complex and needs higher computational time.
- Numerical methods like Newton-Raphson to solve them.

Solvers

Interior point Method

- It(HPIPM) can only be applied to **Quadratic programming(QP)** and **Sequential quadratic problem (SQP)** problems.

$$\text{Minimize: } J(\mathbf{u}) + \varepsilon B(\mathbf{u})$$

$$\text{Where: } B(\mathbf{u}) = \sum_{i=1}^{q} -k_i \ln(-g_i(\mathbf{u}))$$

$$\text{Subject to: } A_2\mathbf{u} - \mathbf{b}_2 = \mathbf{0}$$

HPIPM

Active Set Method

- Active Set(qpOASES) can be applied to **Linear programming (LP)** or **QP** or **SQP** problems.
- Assumes certain **inequality constraint are active** $(\mathrm{A}_{1i}u - b_{1i} = 0)$ and other are inactive.

$$\text{Minimize: } J(u)$$

$$\text{Subject to: } \mathrm{A}_1 u - b_1 = g(u) \leq 0$$
$$\mathrm{A}_2 \mathrm{u} - \mathrm{b}_2 = h(u) = 0$$

$$\text{Optimality: } \nabla J(u^*) + \mu^T \nabla g(u^*) + \lambda^T \nabla h(u^*) = 0$$

qpOASES